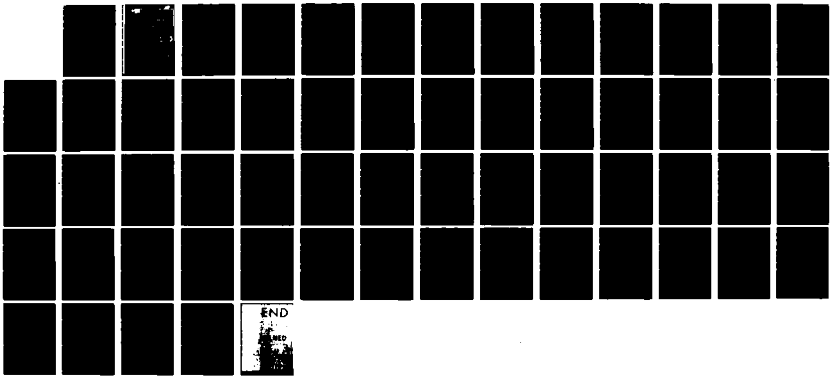
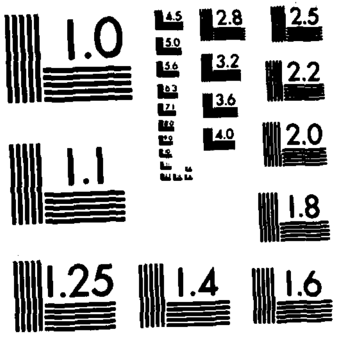


END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

[illegible] NONLIN[illegible]

AN ISOLA[illegible]
FOR THE ANAL[illegible]

Y.S. [illegible]
P.M. Lin

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

April 1983

This work was supported by Office of Naval Research,
Contract No. N00014-81-K-0323.

83 04 19 193

FAULT DIAGNOSIS OF NONLINEAR ANALOG CIRCUITS

VOLUME IV

AN ISOLATION ALGORITHM FOR THE ANALOG FAULT DICTIONARY

Y. S. Elcherif
P. M. Lin

April 1983

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

This work was supported by Office of Naval Research,
Contract No. N00014-81-K-0323.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A127020	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Fault Diagnosis of Nonlinear Analog Circuits, Volume IV, An Isolation Algorithm for the Analog Fault Dictionary.		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Y. S. Elcherif , P. M. Lin		8. CONTRACT OR GRANT NUMBER(s) N00014-81-K-0323
9. PERFORMING ORGANIZATION NAME AND ADDRESS Purdue University School of Electrical Engineering West Lafayette, Indiana 47097		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, Virginia		12. REPORT DATE April 1983
		13. NUMBER OF PAGES 53
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) SAME		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

SAME

18. SUPPLEMENTARY NOTES

NONE

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fault Diagnosis
Fault Dictionary

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A new approach for fault location in an analog fault dictionary has been adopted on the basis of quantizing circuit reponses. The possibility of quantization is offered by faults having nearly the same effect on some test measurements. This produces multi-valued logic responses which can be manipulated logically to obtain decision rules for fault location. A logical isolation algorithm has been introduced to form a dictionary for hard fault diagnosis using d.c. voltage measurements. The dictionary consists of tables containing voltage ranges of quantized response and numerical codes identifying different faults in the dictionary. The test measurements chosen by the algorithm are free from redundancy and provide the maximum fault isolation capability of all initially chosen meansurements. The algorithm can be easily implemented with some extra software added to the circuit simulator used in fault simulation. Hardware implementation of a logical isolation ATE is simple and efficient compared to a least squares based dictionary. The algorithm has been extended to handle multiple test input conditions.

TABLE OF CONTENTS

ABSTRACT

A new approach for fault location in an analog fault dictionary has been adopted on the basis of quantizing circuit responses. The possibility of quantization is offered by faults having nearly the same effect on some test measurements. This produces multivalued logic responses which can be manipulated logically to obtain decision rules for fault location. A logical isolation algorithm has been introduced to form a dictionary for hard fault diagnosis using d.c. voltage measurements. The dictionary consists of tables containing voltage ranges of quantized responses and numerical codes identifying different faults in the dictionary. The test measurements chosen by the algorithm are free from redundancy and provide the maximum fault isolation capability of all initially chosen measurements. The algorithm can be easily implemented with some extra software added to the circuit simulator used in fault simulation. Hardware implementation of a logical isolation ATE is simple and efficient compared to a least squares based dictionary. The algorithm has been extended to handle multiple test input conditions.

1. INTRODUCTION

The problem of analog fault diagnosis of electronic circuits has been addressed by many researchers from different view points [1]. Within the frame of general testing strategy of electronic equipments, we are here concerned with fault location in malfunctioning analog circuit boards. The term "testing strategy" roughly means the procedure followed in testing different modules in modularly structured equipments. For example in case of complete failure, the first module to be checked is the power supply module, which is more likely to be the cause of trouble than any other module, based on trouble shooting experience and fault history. Other types of malfunction may be indicative of the failing module, e.g. if the picture is fading in a monitoring system, the most likely cause may be the video processing unit. In the worst case of a completely unknown source of trouble, automatic fault diagnosis may be applied sequentially to all modules, in which case a fast enough technique has to be available in order to achieve the job in a reasonably short time.

This discussion has been meant to give some insight into practical considerations in trouble shooting which are in direct connection with our approach in this report. The figure of merit which we have planned to achieve is the speed of diagnosis without degrading its reliability, with the purpose of handling a large number of circuits in a short time.

Looking at the different fault diagnosis approaches available in literature [2] we can note two main categories in which every method can approximately be included. These are the "simulation after test" and "simulation before test" approaches. Simulation after test, sometimes called component simulation [3], is intended to solve for component values from knowledge of some network functions (voltages, currents or impedances) measured at accessible network terminals. This is done by solving the network equations in frequency or time domain as derived by KCL, KVL and component characteristics. This implies that all simulations have to be done after actual testing. Adding to this the complexity of the equations to be solved, it turns out that powerful computing facility and long computing time are needed which does not serve the purpose of our work. Naturally we have adopted the other approach in order to minimize the computations needed after testing or completely eliminate them if possible.

This, often called fault dictionary approach, is in its essence an automatic means for replacing human logic in troubleshooting which depends on counting all or most possibilities of likely faults until the true fault is located. This requires solving the network equations under all fault cases, which may be caused by single or multiple component failure, and storing the computed response or any network function which will be taken as a criterion to be compared against measured values after testing. The criteria employed could be voltage, current, etc. The problem that obviously faces this technique is the multitude of faults which must be considered in simulation. However, it has been reported [4] that catastrophic faults (open and short-circuits), also called hard faults, constitute more than 80 percent of encountered faults, which suggests that simulating only these faults, one can locate a good percentage of faults using only an efficient fault dictionary. This saves the effort needed in component simulation every time a diagnosis is needed. With only open and short circuit faults considered, it turns out that only d.c. analysis of the network is needed, which makes fault simulation even simpler.

Organization of the report:

The next section explains the general procedure for fault dictionary set up and presents a least squares based algorithm for fault isolation in the dictionary. Section 3 presents a new algorithm for fault isolation based on quantizing circuit responses. The possibility of quantization is offered by faults having nearly the same effect on some test measurements. The algorithm is built around the digital representation of the responses and is implemented by logical manipulation of the quantized responses. In section 4, the logical binary equivalent form of the algorithm is discussed. This offers a solution to the problem of minimizing the number of test measurements. Important issues in the cost of implementing the algorithm and building a logical isolation-based ATE are discussed in the context of binary representation of the algorithm. In section 5, the algorithm is extended to handle multiple test input conditions. The use of different test inputs is subject to the insufficiency of test measurements under normal operating input conditions. Section 6 contains conclusions and suggestions for further research.

2. AN OVERVIEW ON THE ANALOG FAULT DICTIONARY

2.1 Generation of the Analog Fault Dictionary

The generation of analog fault dictionaries is in fact an iterative process with two degrees of freedom which represent the possibilities of varying the input excitation and selecting the responses to be measured as shown in Fig. 2.1. The selection of the fault list is essentially associated with the original design of the unit under test (UUT). However, an experienced person can replace the designer in selecting the faults which are likely to happen. The selection of the measurable responses is influenced by the required level of isolation and the economy of the testing procedure insofar as the number of measurements is concerned. The input stimuli considered here are d.c. voltage or current sources which may be different from the nominal operating sources and applied at judiciously selected points in the network. The reason behind using any additional input stimulus is the possible insufficiency of the selected measurable responses to achieve the required level of isolation. Applying the additional input, it is

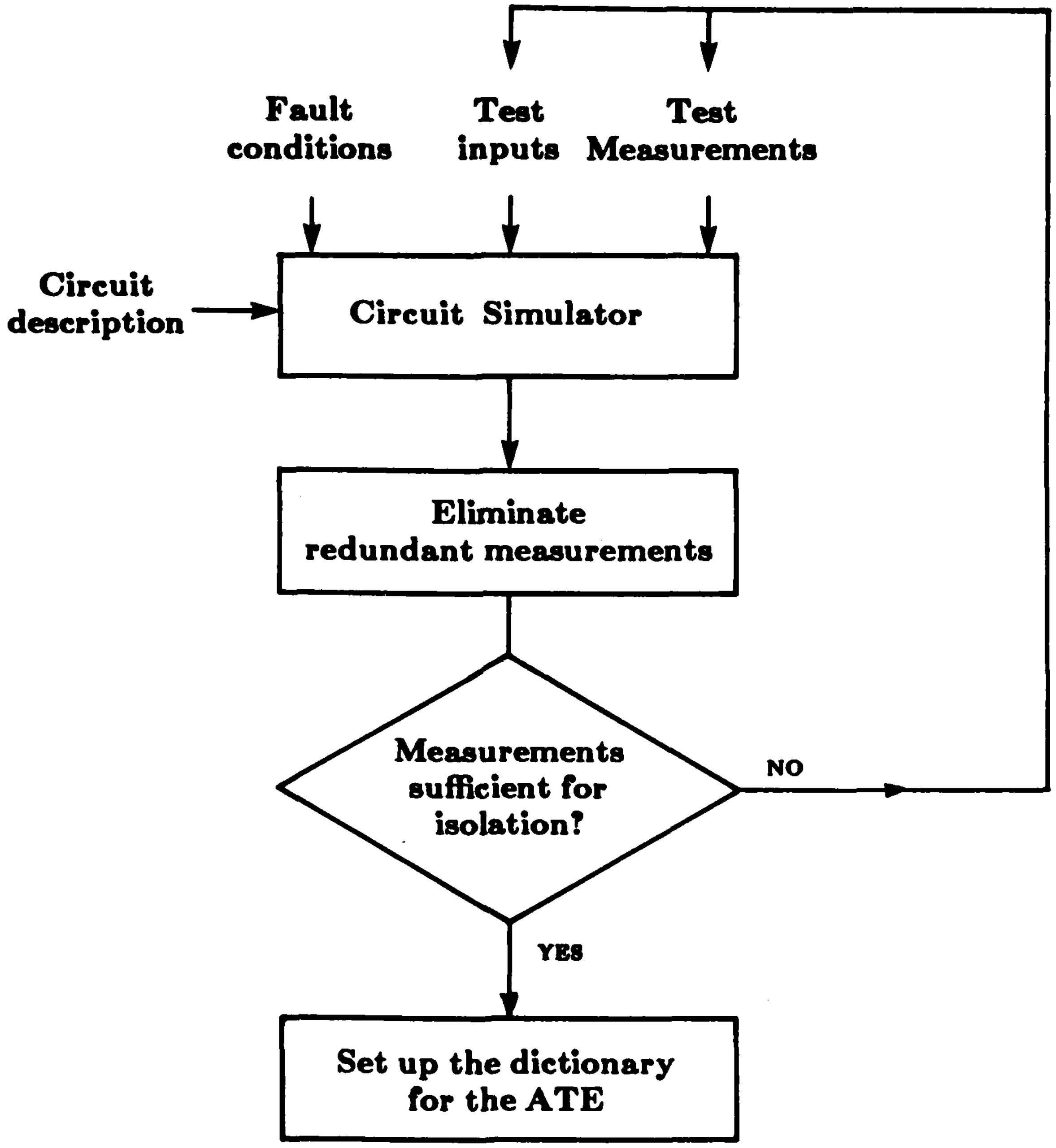

Fault conditions
Test inputs
Test Measurements
Circuit description
Circuit Simulator
Eliminate redundant measurements
Measurements sufficient for isolation?
NO
YES
Set up the dictionary for the ATE

Figure 2.1 Flow chart of the general procedure for generating an analog fault dictionary.

hoped that some of the faults, which have not been isolated with nominal sources, will be propagated to some of the accessible terminals at which responses are measured.

Once the required faults are satisfactorily isolated, the software and data of the automatic test equipment (ATE) have to be prepared. In its basic form, the ATE data is the computed response values which have been found enough for isolation. Other necessary software support depends on how faults will be located in the dictionary after testing, or in other words the decision rule employed in fault location. The capabilities of the circuit simulator are determined by the requirements of the type of analysis to be used. Some circuit analysis programs may not allow direct simulation of open and short circuits without changing the program input in which case, a method for simulating them has to be tailored to suit the program and the method of analysis. The repeated solution of the network under all fault conditions may be time consuming. For this reason, specially designed methods for hard fault simulation have been suggested [5], [6] which do not require solving the original network every time a fault is simulated. In all the discussions to come we will assume that the responses to be used in isolation have been computed already, regardless of the method used in computing them. More specifically, we will focus on d.c. node voltages because of their easy measurement.

At this point it is important to make clear the difference between two problems which are defined below.

1. Fault isolation:

This means finding a set of tests or measurements that enable us to recognize all faults if any one of them happens. Before dealing with the problem, it may be instructive to note some of the inherent features of analog responses which govern any fault isolation study:

1. Unlike digital functions, analog voltages span a continuous range of values where it may seem that the exact value of the voltage is the quantity of interest. However, the evident variability of any response as a result of the components' statistical variation makes it necessary to talk about ranges of voltages instead of their exact values.
2. The network topology imposes some restrictions on the sensitivity of responses measured at accessible terminals to some of the faults, at least under normal operating conditions. In other words, not every fault will affect every terminal. This will result in having groups of faults having almost the same effect on some node voltages. We referred to this earlier as the insufficiency of some terminal measurements to reveal the actual fault. Rather we will have ambiguous cases and ambiguity sets of faults which have to be further analyzed and broken down to their elementary constituents using different measurements. This process is known

in reliability studies as a fault tree [7].

2. Fault Location:

By fault location, we mean the procedure to be followed after testing to locate the actual fault among all faults in the dictionary. The problem may be better viewed in the space of measurements. Suppose that $\{v_1, v_2 ... v_n\} \in \mathbb{R}^n$ is the set of measurements necessary for location of any fault in the set $\{f_1, f_2 ... f_m\}$. The result of simulation is m points in $\mathbb{R}^n$ representing the m faults. Given the actual measurement represented by a point in $\mathbb{R}^n$ it is required to find a point of the m simulations which is nearest to the actual measurement. The difficulty of the problem arises from the fact that the actual measurement will never coincide with any of the simulations because of the statistical variation of the components in the network and consequently of the node voltages. This problem is obviously a pattern classification problem where every class is only a single point in $\mathbb{R}^n$ and the pattern to be classified is also a single point. This is called "the nearest neighbor" problem. The nature of the problem as presented here may not always be explicitly stated in literature dealing with fault dictionary, however we view the decision rule for fault location in a dictionary as a pattern classifier which is obvious in a linguistic sense. Classification or finding the nearest neighbor is done on an optimal basis which may or may not take into consideration the statistical properties of the pattern to be classified. Optimality is usually sought by defining a distance measure which is thought to be best representative of the properties of the space dealt with. For example, in $\mathbb{R}^n$ the second order Euclidean norm is often considered as the optimality criterion.

2.2 Fault Location Based on Minimum Sum of Squared Deviations

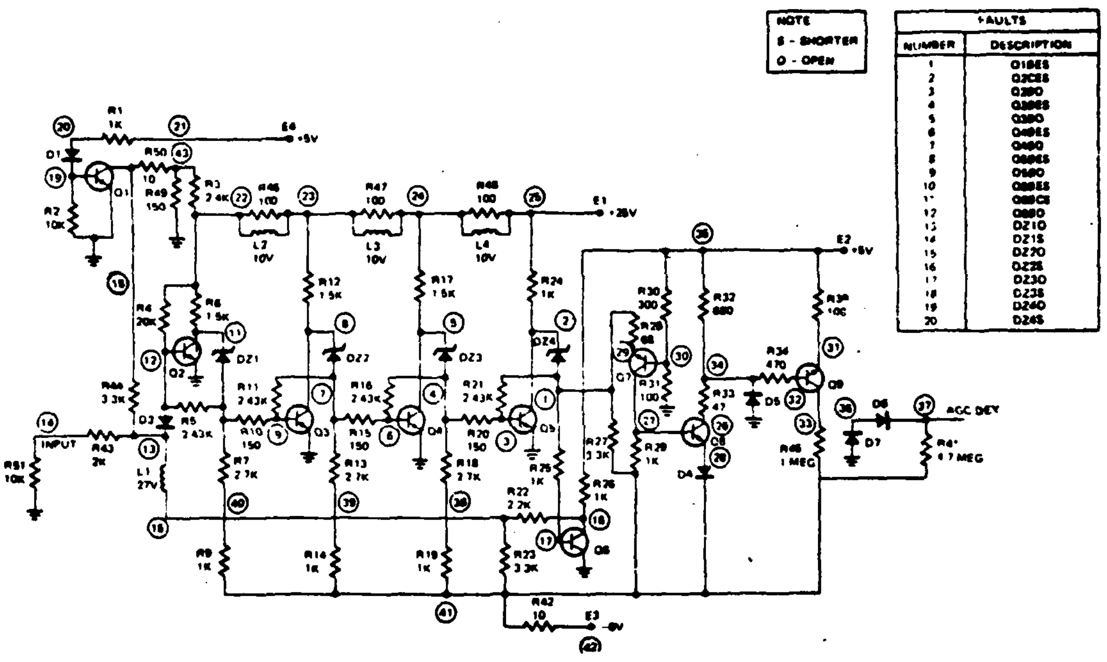
Hochwald and Bastian in [4] described a method for fault location using a d.c. analog fault dictionary. Their study involved an extra step called fault detection. They used SYSCAP II simulator to calculate node voltages under nominal and faulty conditions of the video amplifier of Fig. 2.2.

Fault detection:

Denote by $v_{1(NOM)}, ... v_{n(NOM)}$ the nominal computed node voltages, and by $v_1(f_i), ..., v_n(f_i)$ the computed voltages under fault condition (f_i) of the corresponding nodes. The sum of squared deviations computed as:

$$\sum_{k=1}^{n} (\Delta v_k)^2 = \sum_{k=1}^{n} |v_k(NOM) - v_k(f_i)|^2 \qquad 2.1$$

was found by them to be always greater than 0.5 n (where n is the number of nodes) in case of successful isolation of the fault later in the analysis, which means an average deviation of approximately 0.7 volt. This suggested that a minimum of 0.5 n sum of



FAULTS	
NUMBER	DESCRIPTION
1	Q1BES
2	Q2CES
3	Q2BO
4	Q3BES
5	Q3BO
6	Q4BES
7	Q4BO
8	Q5BES
9	Q5BO
10	Q8BES
11	Q8BCS
12	Q8BO
13	DZ1O
14	DZ1S
15	DZ2O
16	DZ2S
17	DZ3O
18	DZ3S
19	DZ4O
20	DZ4S

Fig. 2.2. Video amplifier discussed in sections 2.2 and 5.

squared deviations is required in order to detect the fault. The only degree of freedom allowed to achieve this for a particular set of arbitrarily chosen test nodes is varying the input stimuli as shown in Fig. 2.3 taken from [4].

Forming ambiguity sets:

The ±0.7 volt average deviation was taken as the range of variation for nominal and simulated faulty voltages. The possibility of overlapping ranges is there. However the ranges themselves are not needed once the faults belonging to every ambiguity set are identified. A problem may exist if some fault exists in the overlap area between two adjacent ranges. However, this was not mentioned, and it seems that human intervention had to be used to select the ranges in such a way to prevent this situation. The result is listed in Table 2.1. The notable thing about this table is the exclusion of the nominal ambiguity set on the basis of the 0.5 n detection criterion. It was believed that faults producing node voltages in the nominal range make the measurement of these voltages useless. We believe that this is a loss of information that caused some unnecessary effort later on during the isolation process. The use of nominal ambiguity sets will prove to be useful in the new algorithm to be explained in the next sections.

Fault isolation:

At this point, it has to be assured that the chosen set of nodes is capable of unique characterization of all faults. It is also important to discover if there are unnecessary measurements which do not help in isolation. The ground rules stated by the authors to check the above two features are:

1. Any ambiguity set which has a single fault within it, uniquely defines that fault at that test node.
2. Ambiguity sets whose intersection or symmetric difference result in a single fault, also uniquely identify the fault.

Indeed the symmetric difference between two sets (of two different node) must be the intersection of one of them with the nominal ambiguity set of the other node which has been omitted.

Starting with the ten nodes in Table 2.1 and using the above mentioned rules it was possible to find a subset of only five nodes (11, 8, 5, 2 and 16) which were sufficient for unique isolation of all faults except the pair 10 and 12. This means isolating 95 percent of the listed 20 faults.

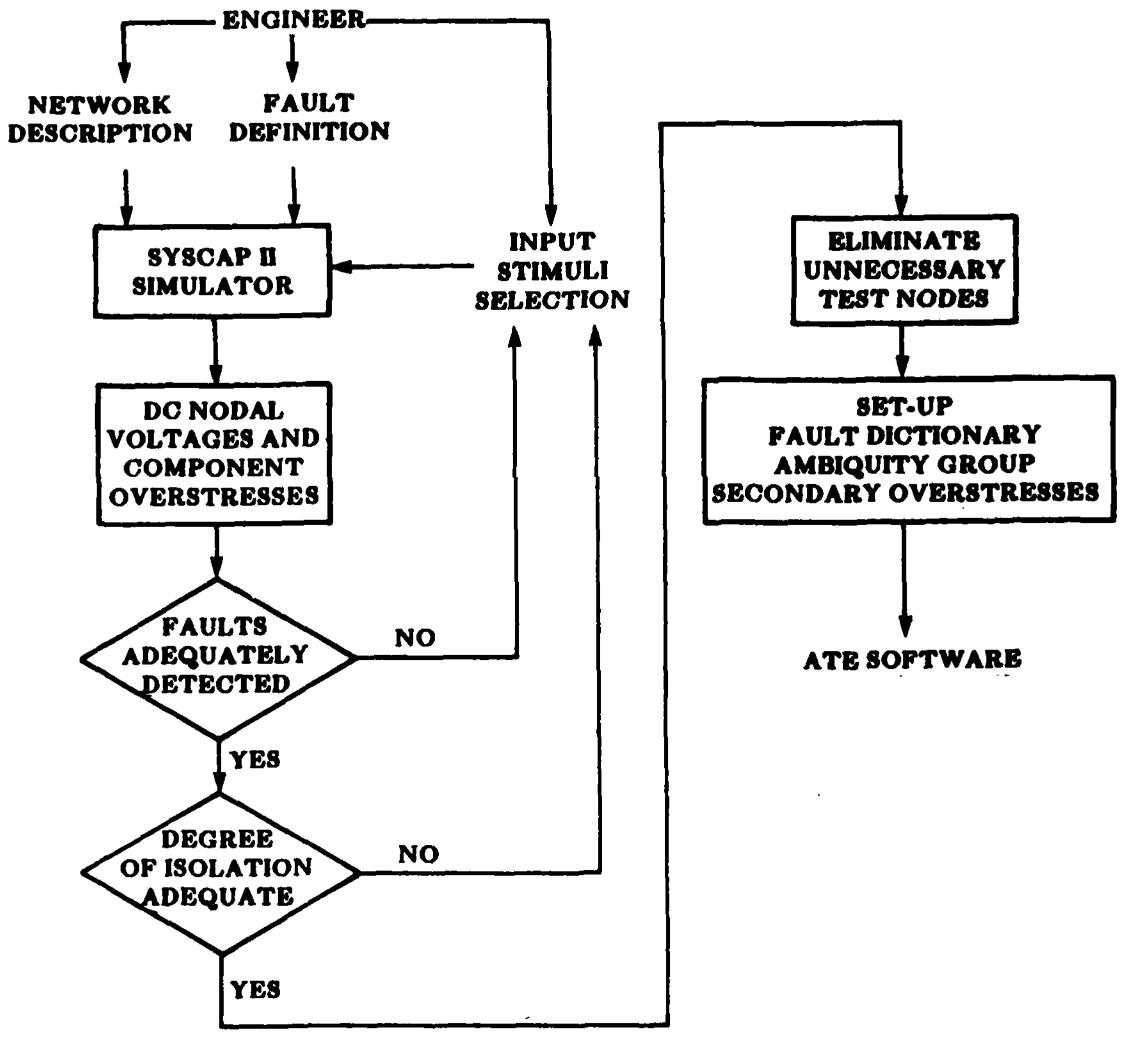

ENGINEER
NETWORK DESCRIPTION
FAULT DEFINITION
SYSCAP II SIMULATOR
INPUT STIMULI SELECTION
DC NODAL VOLTAGES AND COMPONENT OVERSTRESSES
FAULTS ADEQUATELY DETECTED
NO
YES
DEGREE OF ISOLATION ADEQUATE
NO
YES
ELIMINATE UNNECESSARY TEST NODES
SET-UP FAULT DICTIONARY AMBIQUITY GROUP SECONDARY OVERSTRESSES
ATE SOFTWARE

Figure 2.3 Flow chart for generating a least square based fault dictionary using SYSCAP II circuit simulator.

Table 2.1. **Ambiguity sets of faults simulated in the video amplifier of fig. 2.2. The nominal set contains the rest of the 20 faults.**

Node	Input	Set No. 1	2	3	4	5
11	+30	3,6,7,15,19	nominal			
	−30	2	5	13	14	nominal
8	+30	3	7	15	16	nominal
	−30	2,4,5,13	nominal			
5	+30	3,6,7,15	nominal			
	−30	2,4,5,13	9	17	18	nominal
2	+30	3,6,7,15	19	20	nominal	
	−30	2,4,5,8,9,13,17	nominal			
27	+30	3,6,7,15,19	nominal			
	−30	2,4,5,8,9,13,17	nominal			
26	+30	3,6,7,15,19	nominal			
	−30	2,4,5,8,9,13,17	nominal			
33	+30	3,6,7,15,19	nominal			
	−30	2,4,5,8,9,15,17	nominal			
36	+30	nominal				
	−30	nominal				
18	+30	nominal				
	−30	nominal				
16	+30	3,6,7,10,12,15,19	nominal			
	−30	2,4,5,8,9,15,17	11	nominal		

Fault location:

The fault dictionary then consists of the values of the simulated voltages of the selected nodes for both inputs under all fault conditions which results in (5x2x19) 190 real numbers. To illustrate the use of the dictionary three faults were simulated on the circuit board and measurements of the five test nodes were taken. The procedure to be followed afterwards is:

1. Finding the difference between simulation results and measured values.
2. Finding the sum of the squares of these differences under both input conditions.

Table 2.2. Measured node voltages versus node voltage predicted by simulation (M=Measured, P=Predicted and D=Difference)

Node	Simulated Faults						
		Q3BES		Q6BES		DZIS	
	Input	−30	+ 30	−30	+ 30	−30	+ 30
	M	5.38	0.15	6.12	0.3	1.82	0.13
	P	5.32	0.15	5.93	0.2	1.7	0.11
11	D	0.06	0.0	0.19	0.1	0.12	0.02
	M	6.45	6.46	0.06	6.44	0.07	6.46
8	P	5.99	5.99	0.09	5.97	0.06	6.0
	D	0.46	0.47	0.03	0.47	0.01	0.46
	M	0.06	0.06	6.38	0.07	6.38	0.07
	P	0.09	0.09	5.93	0.09	5.92	0.09
5	D	0.03	0.03	0.45	0.02	0.45	0.02
	M	6.73	6.77	0.09	6.85	0.09	6.73
2	P	6.95	6.95	0.12	6.91	0.12	6.95
	D	0.22	0.18	0.03	0.06	0.03	0.22
	M	0.10	0.13	2.26	3.95	2.31	0.13
16	P	0.02	0.03	3.07	4.22	3.11	0.03
	D	0.08	0.10	0.81	0.27	0.8	0.10

Table 2.3. Sum of the squared deviations for three hardware induced voltages.

	Induced Faults								
	Fault 4 Q3BES			Fault 10 Q6BES			Fault 4 DZIS		
Fault No.	−30	+ 30	Total	+ 30	−30	Total	+ 30	−30	Total
1	130	0.3	130	1	16	17	18	0.3	18
2	29	0.3	29	163	16	179	129	0.3	129
3	128	171	299	0.9	154	155	18	171	189
4	0.3	0.3	0.6	128	16	144	139	0.3	139
5	7	0.3	7	132	16	148	167	0.3	167
6	128	103	231	0.9	87	88	18	102	120
7	128	100	228	0.9	85	86	18	99	117
8	69	0.3	69	53	16	69	70	0.3	70
9	109	0.3	109	57	16	73	74	0.3	74
10	129	17	146	0.9	0.3	1.2	18	17	35
11	119	0.5	119	3	11	14	20	0.6	20
12	129	17	146	0.9	0.3	1.2	18	17	35
13	93	0.3	93	127	16	143	300	0.3	300
14	141	0.3	141	20	16	36	0.9	0.3	1.2
15	129	177	306	0.9	160	161	18	178	196
16	129	22	151	0.9	38	39	18	22	40
17	264	0.3	264	126	16	142	143	0.3	143
18	96	0.3	96	23	16	39	40	0.3	40
19	129	86	215	0.9	68	69	18	86	104
20	129	19	148	0.9	35	36	18	18	36

3. Finding the smallest number in these sums.

The actual fault is expected to produce the smallest sum of boxes. The results of the above three steps are shown in Tables 2.2 and 2.3 with the minimum sum of squares enclosed in squares. The fault pair 10 and 12 which could not be isolated are seen to give the same sum of squares.

3. A NEW ALGORITHM

The basis of the new algorithm is making full use of the ambiguity clusters of faults which have almost the same effect on terminal measurements. The result is to confine our interest only to the ranges of voltages occupied by measurements of a faulty circuit instead of the whole continuum of values. This effectively transforms the problem from analog to digital. However this digitization yields multivalued logical response instead of the usual binary. It is easy to see that this can be further transformed to binary logic if every set of faults is treated as a binary variable. If the response lies within the range (v_{min}, v_{max}) a logical 1 is obtained. Otherwise it gives a logical zero. This is shown in Fig. 3.1. The isolation process can be looked at as a logical realization problem whether it be in binary or multilevel form. We will first present the solution in the multilevel form for better clarity, then proceed to consider important implications of the binary representation. The requirement of the realization problem is to obtain a digital expression for every fault (as a digital function) in terms of the digital variables which are the ambiguity sets. A systematic fault tree generation will be shown next, which simultaneously achieves the following:

1. Eliminate the redundancy in the test measurements.
2. Find out the maximum isolation capability of the initially chosen set of measurements.
3. Generate the required logical expressions of the faults. These logical expressions will form the required dictionary. The arrangement of the expressions in the dictionary will make the fault location task fairly easy and efficient. To prevent confusion, we should note here that the logical expression will be described often as "fault code". We will also use the word "set" sometimes instead of ambiguity set.

3.1 Forming Ambiguity Sets

Classifying the faults into ambiguity sets depends on some tolerance of the node voltages. In the example to follow in this section a tolerance of ±0.7 volts is used. The faults are divided into ambiguity groups which cause the node voltage to be in corresponding ranges. Each range is centered around the voltage value due to some fault called the center of the ambiguity set. Eventually, each node will produce different grouping of the faults depending on how the faults affect this particular node

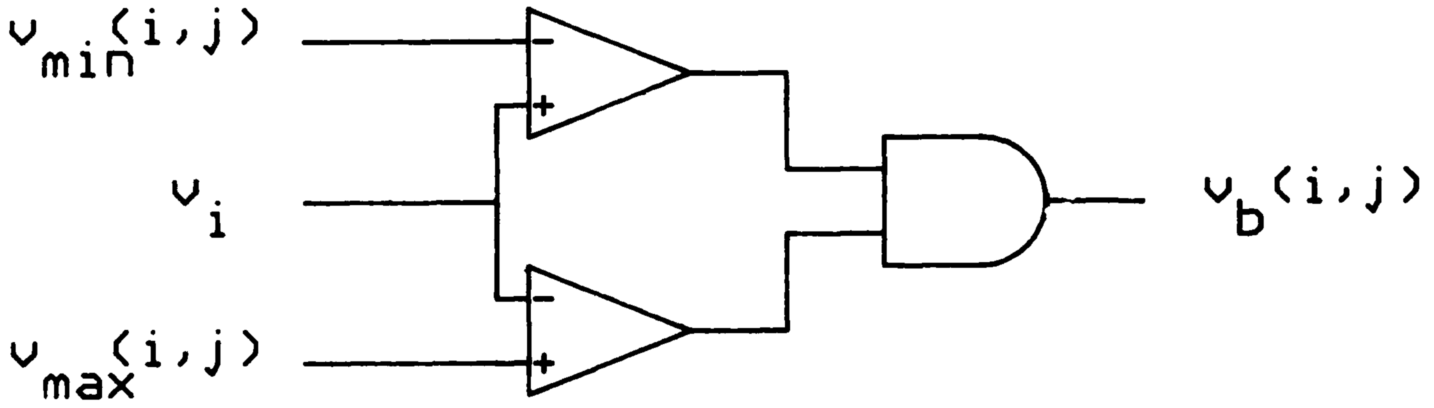


Fig 3.1 Generation of ambiguity set binary variables $\{v_b(i,j)\}$

voltage. Thus there will be several ambiguity sets of faults, or just one set, associated with each node. Since there is no priority given to any particular fault, the ambiguity sets of every node can be found from "the node" voltages under different fault conditions by the following procedure:

1. Start with the nominal case as the center of ambiguity set j, $j = 1$ (called nominal set).
2. Scan all faults. If any voltage is within ±0.7 volt of the center voltage, include the corresponding fault in the set j, $j = 1$.
3. When all faults are scanned, take the first fault which has not been included in a previous set to be the center of the new set and let $j = j+1$. If every fault has been included in some set stop.
4. Do the same as step 2 for the current set j. If an overlap occurs between the range of this set and any previous range, divide the overlap region into two halves. Any voltage in the overlap region will cause the corresponding fault to belong to the set whose center is nearer. Go to step 3.

3.2 Fault isolation

Given the ambiguity sets derived from the simulation results, the algorithm proceeds as follows. Integer values are assigned to the ambiguity sets of every node in the set of nodes $\{v_i\}_1^{N_v}$, where N_v is the number of test nodes. The different nodes will generally possess different numbers of ambiguity sets. The ith node has L_i sets including the nominal ambiguity set (i.e. faults which do not cause deviation from the nominal response).

A fault is characterized by being the single element contained in any set or the intersection of any number of sets of different nodes (since any fault belongs only to one set of every node). Therefore, intersection operations will be performed. The object to be achieved is to have a number of sets resulting from intersection which is equal to the number of faults N_f. In this case, all faults will be isolated since every set cannot contain more than one fault and in the same time every fault cannot be contained in more than one set.

If all sets of all nodes are exhausted without achieving N_f nonempty intersections, the result will be the best degree of isolation that can be obtained with the available nodes. In this case, some sets resulting from intersection will still contain more than one fault. These contents cannot be distinguished using only these nodes without additional input stimuli.

If it happens after intersecting the sets of some node with the resulting sets of previous intersections that the total number of sets is not increased, then no information is added by this new node. This means that this node is redundant and has to be excluded. This would imply that there is no redundancy in the set of nodes obtained after intersection, but it would not guarantee that the number of nodes is minimum, i.e. there is no subset of these nodes which could achieve the same degree of isolation, but there could be another set of other nodes, less in number, and same in isolation capability. This depends on the sequence of nodes being intersected. However, to approach the minimum number of nodes, one should give priority to nodes with higher isolation capability, which is in this case represented by a larger number of ambiguity sets.

The exact minimum number of nodes may not be of considerable practical importance. The minimization problem will be presented later in the context of Boolean representation of the isolation process. The solution to the problem, though known, is time consuming. The isolation process is illustrated in the fault tree diagram of Fig. 3.2.

3.3 Generation of fault codes:

The third goal of the algorithm which is generating the fault codes is done using a "labeling technique" (after a well-known graph algorithmic notion [8]) which employs the integer values assigned to the different logic levels (i.e. the different ambiguity sets). The method is basically describing every fault in terms of the sets in which this fault is a common member. This is done by identifying every set resulting from intersection by the labels of the sets producing it, in proper order. After the last intersection step, we will have an integer code characterizing every fault. The length of the code is equal to the number of nodes. The implementation of this part requires a stack (an expandable storage) which is updated after every intersection. This will be best illustrated through the example treated next. A flow chart of the whole isolation process is shown in Fig. 3.3.

The algorithm:

Denote by v(i,j) the ambiguity set j of node i (it will be used to express both the voltage range and the fault contents). For example, v(1,3) refers to ambiguity set 3 of the first node which is VT2. Let the number of nodes be N_v, the number of faults be N_f and the number of ambiguity sets in the $i^{\underline{th}}$ node be L_i. Create two 2-dimensional arrays x(l) and y(l) of capacity $N_f x N_f$ integer storage locations, so that we have N_f x's and y's accounting for a maximum number of N_f sets resulting from intersection. Each x or y should accommodate a maximum number of N_f integers accounting for the faults. The index referring to the fault number has been omitted. Therefore, the index l in x(l) denotes the lth set resulting from intersection of other sets. Also create an $N_f x N_v$ two-dimensional stack s(l,n) such that each of the N_f registers written vertically in a column can store up to N_v integer numbers in a row. This stack will eventually contain the required fault codes while the corresponding faults will be in the x storage. The y storage is for temporary holding of the intersection results. The algorithm then

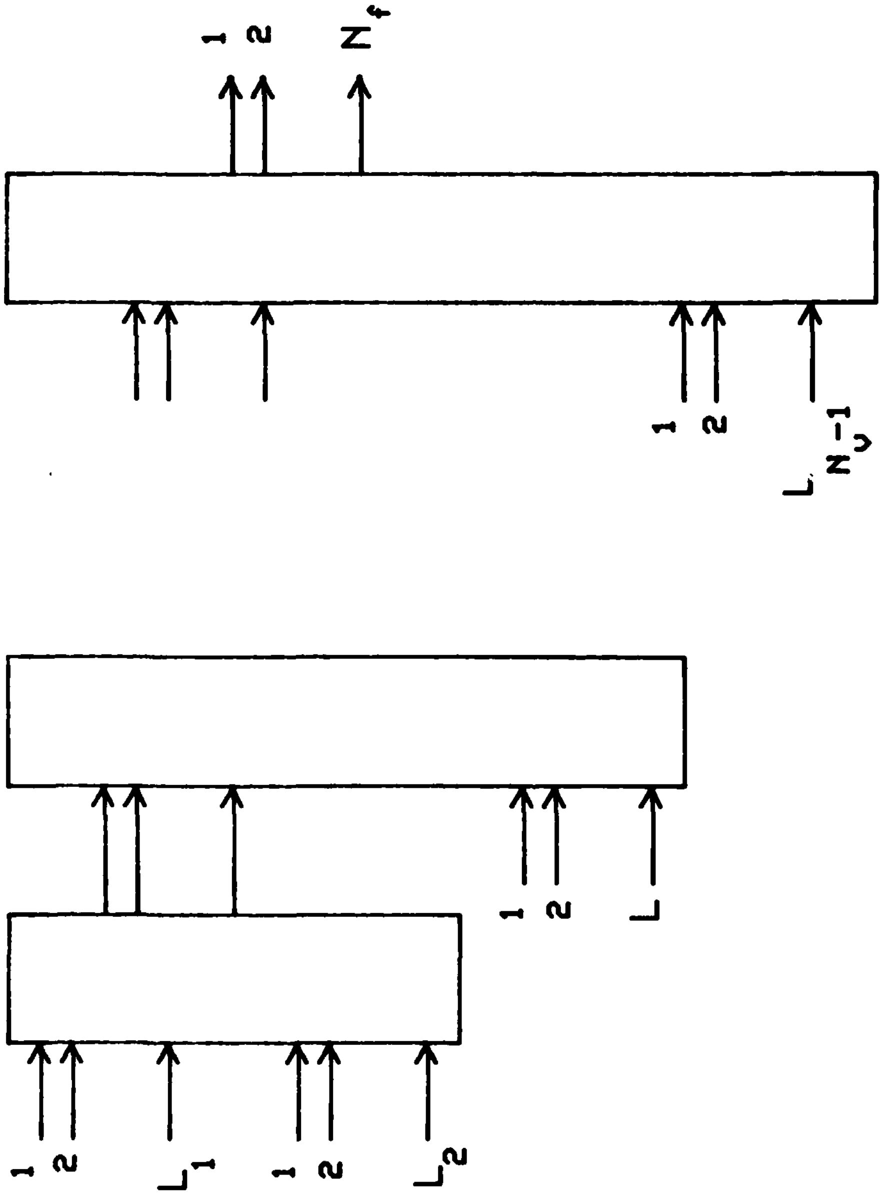


Fig 3.2 Fault tree generation by intersection of ambiguity sets

proceeds as follows:

1. Let $n = 1$ as an index for the number of nodes.
2. Find the node i_1 that has the maximum number of sets L_1. Load the first column of the stack with the ambiguity set numbers of node i_1:

$$s(j,1) = j \qquad j = 1,\ldots,L_1 \tag{3.1}$$

   And load x(j) with the contents (faults) of set j.
3. Let $n = n+1$. If $n > N_v$ stop. Otherwise find the node i_2 that has the next maximum number of sets L_2.
4. Let $l = 1$, $j = 1$ and $k = 1$.
5. Take the intersection:

$$y(l) = x(j) \cap v(i_2,k) \tag{3.2}$$

   where $v(i_2k)$ contains the faults of the ambiguity set k of the node i_2.
6. If $y(l) = \phi$ take the next set $v(i_2,k)$, $k = k+1$ or if the sets of node i_2 are exhausted take the next set x(j), $j = j+1$, then set $k = 1$ and go to 5.
   If $y(l) \neq \phi$ and $l = 1$ (i.e. the first intersection of the set x(j)) skip the following stack updating and go to 7.
   If $y(l) \neq \phi$ and $l \neq 1$ update the stack contents as follows:

$$s(l,m) = s(l-1,m) \qquad m = 1,\ldots,(n-1) \tag{3.3}$$

7. Add the set index k of the current node i_2:

$$s(l,n) = k \tag{3.4}$$

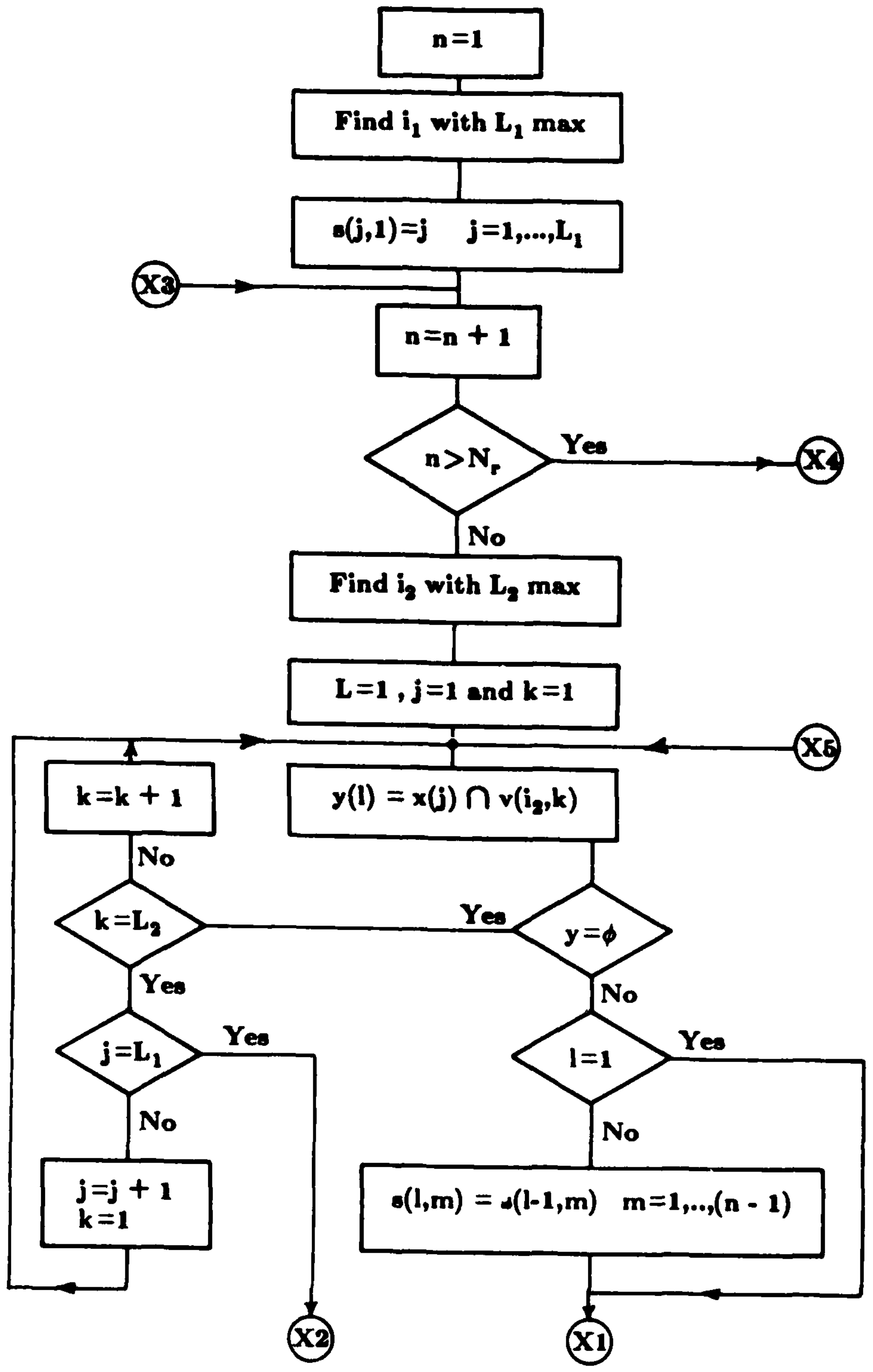
8. If $l < N_f$ let $l = l+1$, $k = k+1$ and go to 5. If $l = N_f$ stop. If all intersections have been made while $l = L_1$, decrement the node index $n = n-1$ and go to 3 to consider a new node. This conditions will happen if the node i_2 is redundant and does not help breaking any ambiguity, thus yielding the same number of sets. Note also that:

$$l \leq N_f \tag{3.5}$$

$$L_1 \leq l \tag{3.6}$$

$$L_2 \leq L_1 \tag{3.7}$$

   If all intersections have been mode yielding $N_f > l > L_1$, move the result of intersection from y to x and go to 3 to get a new node.



n=1
Find i_1 with L_1 max
s(j,1)=j   j=1,...,L_1
X3
n=n + 1
n>N_r
Yes
X4
No
Find i_2 with L_2 max
L=1 , j=1 and k=1
X5
k=k + 1
y(l) = x(j) ∩ v(i_2,k)
No
k=L_2
Yes
y=ϕ
Yes
No
j=L_1
Yes
l=1
Yes
No
No
j=j + 1
k=1
s(l,m) = s(l-1,m)   m=1,..,(n - 1)
X2
X1

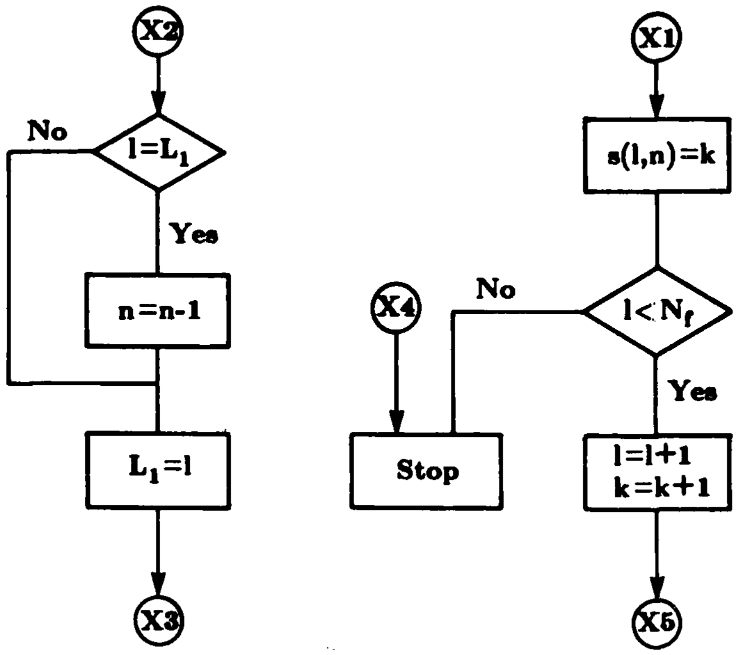


Figure 3.3 Flow chart of the isolation algorithm.

Example 3.1

The video amplifier shown in Fig. 3.4 has been analyzed using a program for hard fault simulation which employs the method of complementary pivot theory described in [6]. The equivalent circuit is shown in Fig. 3.5. The faults simulated are listed in Table 3.1. Initially eleven test nodes were chosen and the node voltages were computed by the program. The result is shown in Table 3.2 for the nominal and the faulty cases where the nominal case is assigned the number 1 among other faults. The program divides the faults into ambiguity groups, each group centered around some fault with ±0.7 v range on each side of the response due to this fault. In case an overlap occurs, the overlap region is halved between the two overlapping ranges. If any fault happens to be in the overlap region, it is included in the set whose center fault is closer to this fault. The result of forming the ambiguity sets and finding the ranges of the sets is shown in Table 3.3 with every set assigned an integer number.

As an example consider the classification of faults into ambiguity sets according to the simulation results in table 3.2, and particularly consider the first node in the table which is VT2:

Set 1:

Center fault: 1 (nominal case)

$$\begin{aligned}\text{voltage range} &= 1.211 \pm 0.7 \\ &= 0.511 \rightarrow 1.911\end{aligned} \tag{3.8}$$

searching in the table for faults which cause the voltage of VT2 to be in the above range, they turn out to be faults 1, 4, 6, 8, 10,11, 13, 14 and 16.

Set 2:

Searching for the first fault which has not been included in the nominal set, fault 2 is taken to be the center of set 2. The coincidence between the center fault and set numbers in this set and the previous one is merely an accident.

$$\begin{aligned}\text{voltage range} &= 5.81 \pm 0.7 \\ &= 5.11 \rightarrow 6.51\end{aligned} \tag{3.9}$$

Comparing the range of set 2 to that of set 1, we find that there is no overlap and no modification is needed. Faults which cause the voltage of VT2 to be in the above range are 3 and 5 in addition to the center fault 2.

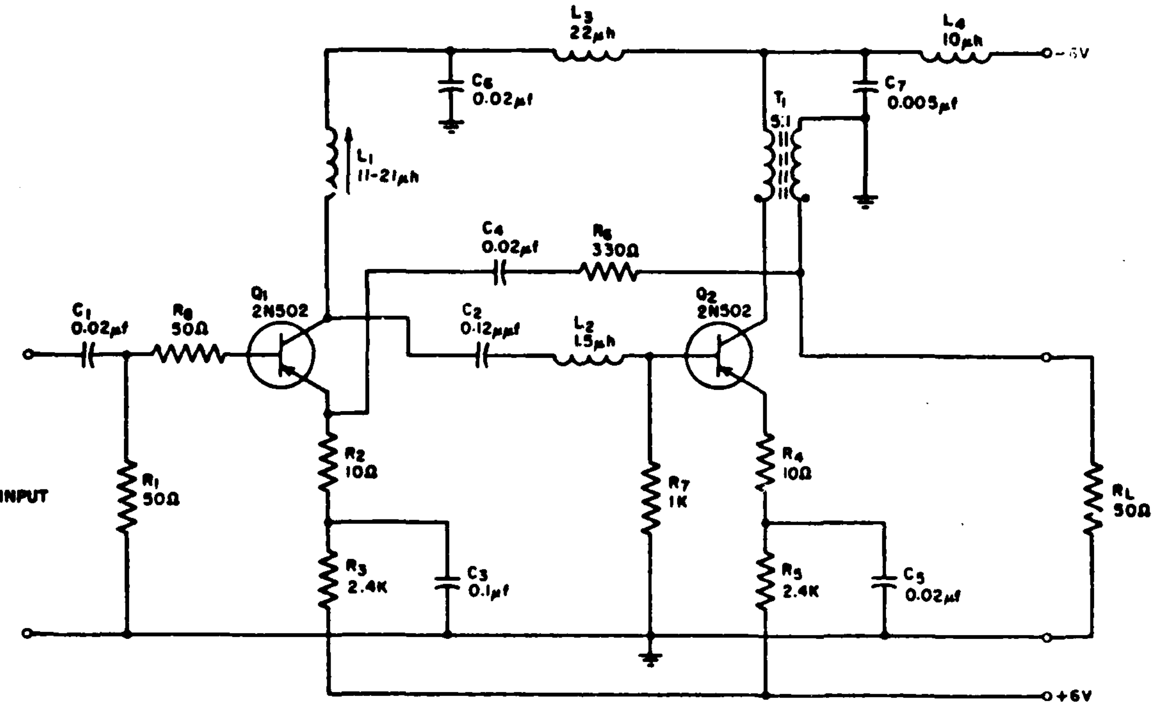


Fig. 3.4. Video amplifier of example 3.1.

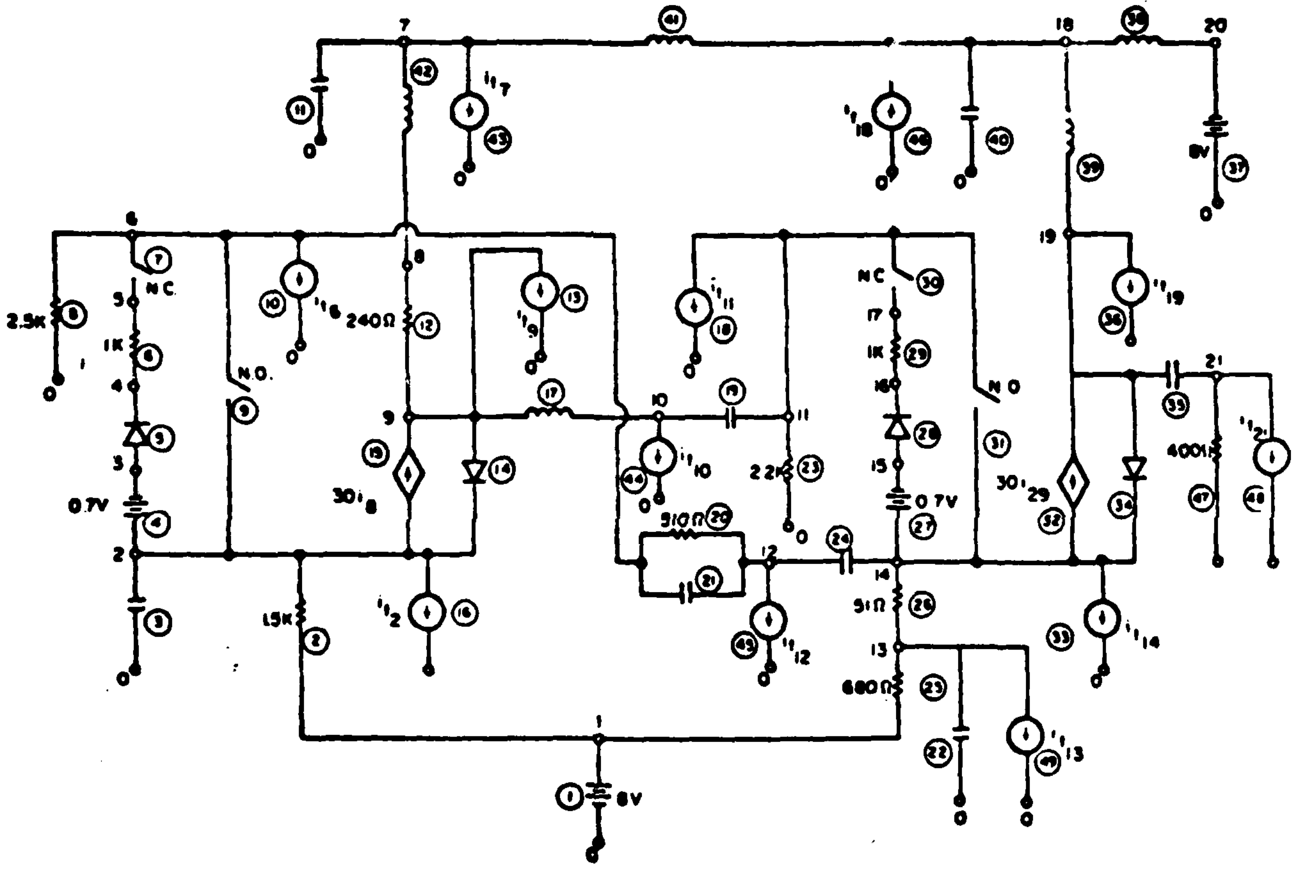


Fig. 3.5. Piecewise linear model of the amplifier in example 3.1.

Table 3.1. Definition of Faults.

number	description
1	nominal case
2	L1 and/or L2 open
3	L4 open
4	L3 open
5	L7 open
6	L5 and/or L6 open
7	Q1 base open
8	Q2 base open
9	C4 shorted
10	C6 shorted
11	C3 shorted
12	C5 shorted
13	C8 shorted
14	C9 shorted
15	Q1, B-E shorted
16	Q2, B-E shorted

```
FAULT-VT TABLE

FAULT NO  1          2          3          4          5          6          7          8          9          10

VT 2       1.211E+00  5.810E+00  5.810E+00  1.211E+00  6.274E+00  1.211E+00  8.000E+00  1.211E+00  2.827E-14  1.211E+00

VT 6       3.650E-01  3.650E+00  3.650E+00  3.650E-01  3.981E+00  3.650E-01  5.684E-14  3.650E-01  1.112E-12  3.650E-01

VT 7      -8.000E+00 -8.000E+00  5.810E+00 -8.000E+00  6.380E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT 9      -6.949E+00  5.810E+00  5.810E+00 -6.949E+00  6.274E+00 -6.949E+00 -8.000E+00 -6.949E+00 -8.000E+00 -6.142E+00

VT11       6.210E-01  6.210E-01  6.210E-01  6.210E-01  3.905E+00  4.085E+00  6.210E-01  9.095E-13  6.210E-01 -6.142E+00

VT12       3.650E-01  3.650E+00  3.650E+00  3.650E-01  3.981E+00  3.650E-01  5.684E-14  3.650E-01  1.112E-12  3.650E-01

VT13       2.050E+00  2.050E+00  2.050E+00  2.050E+00  6.493E+00  6.737E+00  2.050E+00  8.000E+00  2.050E+00 -3.976E+00

VT14       1.603E+00  1.603E+00  1.603E+00  1.603E+00  6.380E+00  6.643E+00  1.603E+00  8.000E+00  1.603E+00 -4.874E+00

VT18      -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00  6.380E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT19      -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00  6.380E+00  6.643E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT21       0.         0.         0.         0.         0.         0.         0.         0.         0.         0.

FAULT NO  11         12         13         14         15         16

VT 2       1.211E+00  2.179E+00  1.211E+00  1.211E+00  5.000E+00  1.211E+00

VT 6       3.650F-01  1.353E+00  3.650E-01  3.650E-01  5.000E+00  3.650E-01

VT 7      -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT 9      -6.949E+00 -7.099E+00 -6.949E+00 -6.949E+00 -8.000E+00 -6.949E+00

VT11       6.210E-01  5.951E-01  1.137E-12  6.210E-01  6.210E-01  6.005E+00

VT12       3.650E-01  1.566E+00  3.650E-01  3.650E-01  5.000E+00  3.650E-01

VT13       2.050E+00  2.015E+00 -5.655E-27  2.050E+00  2.050E+00  6.144E+00

VT14       1.603E+00  1.566E+00 -7.958E-13  1.603E+00  1.603E+00  6.005E+00

VT18      -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT19      -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00 -8.000E+00

VT21       0.         0.         0.        -8.000E+00  0.         0.
```

Table 3.2. Node voltages of the amplifier in example 3.1 under 16 fault cases.

Set 3:

The center fault of set 3 is the first fault which has not been included in any of the two previous sets. This turns out to be fault 7.

$$\begin{aligned}\text{voltage range} &= 8.0 \pm 0.7 \\ &= 7.3 \rightarrow 8\end{aligned} \tag{3.10}$$

The upper voltage of the range is taken to be 8.0 not 8.7 because the maximum supply voltage is 8.0 volts. A search in Table 3.2 indicates that fault 7 is the only one in this set. The voltage range does not overlap with previous ranges.

Set 4:

Center fault: 9

$$\begin{aligned}\text{voltage range} &= 0.0 \pm 0.7 \\ &= -0.7 \rightarrow 0.7\end{aligned} \tag{3.11}$$

By comparison to previous ranges, we find that this range overlaps with the range of set 1. Henceforth modification is needed to lower the upper voltage of this range and raise the lower voltage of the range of set 1. This results in

voltage range of set 4: -0.7 → 0.605

voltage range of set 1: 0.605 → 1.911

Fault 9 turns out to be the only fault in set 4.

Set 5:

Center fault: 12

$$\begin{aligned}\text{voltage range} &= 2.179 \pm 0.7 \\ &= 1.479 \rightarrow 2.879\end{aligned} \tag{3.12}$$

No overlap exists with any previous range, and the set contains no other fault beside fault 12.

Set 6:

Center fault: 15

voltage range = 5 ± 0.7

= 4.3 → 5.7

Modification of the overlapping ranges of this set and of set 2 results in:

$$\begin{aligned}\text{voltage range of set } 6 &= 4.3 \rightarrow 5.345 \\ \text{voltage range of set } 2 &= 5.345 \rightarrow 6.51\end{aligned} \tag{3.13}$$

No other faults exist in set 6.

Table 3.3. Ambiguity sets of example 3.1.

Node	Set	Center fault	Range from	Range to	faults
VT2	1	1	0.605	1.911	1,4,6,8,10,11,13,14,16
	2	2	5.345	6.51	2,3,5
	3	7	7.3	8.0	7
	4	9	-0.7	0.605	9
	5	12	1.479	2.879	12
	6	15	4.3	5.345	15
VT6	1	1	-0.335	1.065	1,4,6,7,8,9,10,11,13,14,16
	2	2	2.95	4.325	2,3,5
	3	12	6.534	2.053	12
	4	15	4.325	5.7	15
VT7	1	1	-8.0	-7.3	1,2,4,6,7,8,9,10,11,12,13,14,15,16
	2	3	5.11	6.51	3,5
VT9	1	1	-7.47	-6.55	1,4,6,8,10,11,12,13,14,16
	2	2	5.11	6.51	2,3,5
	3	7	-8.0	-7.47	7,9,15
	4	10	-6.55	-5.44	10
VT11	1	1	-0.079	1.321	1,2,3,4,7,8,9,11,12,13,14,15
	2	5	3.205	4.605	5,6
	3	10	-6.842	-5.442	10
	4	16	5.305	6.705	16
VT12	1	1	-0.335	1.065	1,4,6,7,8,9,10,11,13,14,16
	2	2	2.608	4.325	2,3,5
	3	12	0.865	2.608	12
	4	15	4.325	5.7	15
VT13	1	1	1.35	2.75	1,2,3,4,7,9,11,12,14,15,16
	2	5	5.79	7.19	5,6
	3	8	7.3	8.0	8
	4	10	-4.676	-3.276	10
	5	13	-0.7	0.7	13
VT14	1	1	0.903	2.303	1,2,3,4,7,9,11,12,14,15,16
	2	5	5.68	7.08	5,6
	3	8	7.3	8.0	8
	4	10	-5.574	-4.174	10
	5	13	-0.7	0.7	13
VT18	1	1	-8.0	-7.3	1,2,3,4,6,7,8,9,10,11,12,13,14,15,16
	2	5	5.68	7.08	5
VT19	1	1	-8.0	-7.3	1,2,3,4,7,8,9,10,11,12,13,14,15,16
	2	5	5.68	7.08	5,6
VT21	1	1	-0.7	0.7	1,2,3,4,5,6,7,8,9,10,11,12,13,15,16
	2	14	-8.0	-7.3	14

The intersection process is started by the two nodes having the maximum numbers of sets which are VT2 (6 sets) and VT13 (5 sets) as shown in Table 3.4. The result of intersection is eleven nonempty sets. The faults in those new sets are listed in the matrix location identified by the row and the column corresponding to the generating sets of VT2 and VT13 respectively. The fault code at this stage consists of only two digits denoting the numbers of the intersecting sets of the corresponding nodes. This is shown in Table 3.5 in an array form before and after intersection. The stack contents before intersection consisted only of the indices of the ambiguity sets of VT2 in the corresponding column. After intersection with the sets of VT13 the stack was expanded to accommodate the 11 sets. The stars to the left indicate that the contents of the stack were pushed down at this position. This happened when an original set had more than one intersection with the new sets in which case the original code had to be repeated. The sequence of nodes considered in intersection is VT2, VT13, VT14, VT6, VT9, VT11, VT12, VT7, VT18, VT19 and VT21. The stack contents and the faults in the corresponding ambiguity sets after intersection with every irredundant node are shown in Table 3.6. The number of sets after intersection with nodes 14, 6, 9, 12, 18 and 19 did not increase. Therefore these nodes were excluded from the dictionary.

The final table of fault codes (Table 3.7) has two faults (4 and 11) unisolated. Referring to the fault list in Table 3.1, these turn to be L_3 open and C_3 short. In such cases the degree of isolation may be accepted and each of the two components may be checked individually after the test if measuring the five nodes 2, 13, 11, 7 and 21 yields the nominal d.c. response, (Note that faults 4 and 11 are in the nominal ambiguity set) which does not add much more effort. In other cases of more ambiguity, d.c. input stimuli may have to be used to reduce the ambiguity as will be seen in a later section.

3.4 Fault location

To illustrate the use of the dictionary, suppose that we have the following measurements obtained from a faulty video amplifier:

$$VT2=1.2\ ,\ V13=8.0,\ V11=0.5,\ V7=-8.0 \text{ and } V21=-8.0$$

Referring to Table 3.2, we find that the fault is in ambiguity set 1 of node 2, set 3 of node 13, set 1 of node 11, set 1 of node 7 and set 1 of node 21. The integer code is then 13111. Now referring to Table 3.7 we find that this corresponds to fault 8. Further referring to Table 3.1 we see that it means transistor Q_2 base open.

Table 3.4. Intersection of the ambiguity sets of nodes VT2 and VT13.

			VT13				
	Set		1	2	3	4	5
		Faults	1,2,3,4,7, 9,11,12,14, 15,16	5,6	8	10	13
	1	1,4,6,8, 10,11,13 14,16	1,4,11,14, 16	6	8	10	13
	2	2,3,5	2,3	5	ϕ	ϕ	ϕ
VT2	3	7	7	ϕ	ϕ	ϕ	ϕ
	4	9	9	ϕ	ϕ	ϕ	ϕ
	5	12	12	ϕ	ϕ	ϕ	ϕ
	6	15	15	ϕ	ϕ	ϕ	ϕ

Table 3.5. Stack contents before and after intersection of the sets of nodes 2 and 13.

Intersection	Stack	X storage
index (l)	VT2	faults
1	1	1,4,6,8,10,11,13,14,16
2	2	2,3,5
3	3	7
4	4	9
5	5	12
6	6	15

Before

Intersection	Stack		X storage
index (l)	VT2	VT13	faults
1	1	1	1,4,11,14,16
2	*1	2	6
3	*1	3	8
4	*1	4	10
5	*1	5	13
6	2	1	2,3
7	*2	2	5
8	3	1	7
9	4	1	9
10	5	1	12
11	6	1	15

After

Table 3.6. Stack contents and isolated faults after intersection with irredundant nodes.

Intersection	Stack			X storage
index (I)	VT2	VT13	VT11	Faults
1	1	1	1	1,4,11,14
2	*1	1	4	16
3	1	2	2	6
4	1	3	1	8
5	1	4	3	10
6	1	5	1	13
7	2	1	1	2,3
8	2	2	2	5
9	3	1	1	7
10	4	1	1	9
11	5	1	1	12
12	6	1	1	15

a) After intersection with VT11

Intersection	Stack				X storage
Index (I)	VT2	VT13	VT11	VT7	Faults
1	1	1	1	1	1,4,11,14
2	1	1	4	1	16
3	1	2	2	1	6
4	1	3	1	1	8
5	1	4	3	1	10
6	1	5	1	1	13
7	2	1	1	1	2
8	*2	1	1	2	3
9	2	2	2	2	5
10	3	1	1	1	7
11	4	1	1	1	9
12	5	1	1	1	12
13	6	1	1	1	15

b) After intersection with VT7

Table 3.7. Fault Dictionary of Example 3.1.

Fault Code (V2,V13,V11,V7,V21)	Fault number
1 1 1 1 1	1,4,11
1 1 1 1 2	14
1 1 4 1 1	16
1 2 2 1 1	6
1 3 1 1 1	8
1 4 3 1 1	10
1 5 1 1 1	13
2 1 1 1 1	2
2 1 1 2 1	3
2 2 2 2 1	5
3 1 1 1 1	7
4 1 1 1 1	9
5 1 1 1 1	12
6 1 1 1 1	15

4. BINARY REPRESENTATION OF THE ALGORITHM

The better insight into the problem which we will have through its binary aspect will reflect into easier implementation of the software of the isolation algorithm and the hardware of the ATE. It will also provide the solution to the problem of minimizing the number of test nodes. The binary nature of the problem arises from two basic features:

1. The measured response will either be in a particular range or not.
2. If a response is found to be in a certain range, a fault will either be in the ambiguity set identified by this range or not, (i.e. it could be one of the possible faults to have caused the trouble or not.)

   The first feature has been shown to convert the ambiguity set range to a binary variable which takes the value one if the response lies within the range and takes the value zero otherwise as shown in Fig. 3.1.

4.1 Ambiguity Set-Fault Truth Table

In the same sense, every fault can be considered as a binary variable which is a function of all ambiguity sets of all the n test nodes. Every fault will take the value one exactly n times when the binary variables representing the corresponding ambiguity sets containing the fault are equal to one. Any other combination of binary values for the ambiguity sets will result in a value zero. In Table 4.1 a truth table is shown for the m binary functions representing the faults in terms of all the binary variables representing the ambiguity sets. The number of these binary variables is equal to $N = \sum_{i=1}^{n} L_i$, where L_i is the number of ambiguity sets of node i. The binary variable representing ambiguity set k of node i is written as v(i,k). The term denoted by this set means that $v_b(i,k)$ takes the value 1 while the other sets assume an unknown combination. This combination itself is actually immaterial as will be shown in the isolation process. The total number of possible combinations of the N variables {(v(i,k)} is equal to 2^N. However N terms are only shown because the rest (2^N-N) terms cannot happen. They cannot happen because the combination of sets is determined by the faults which have possibly occurred, and the reduced truth table give all such possibilities. The "can't happen terms" are useful in obtaining minimal realization.

The problem of fault isolation can now be stated in terms of binary function realization. It is required to synthesize a combinational binary logic whose input is the binary variables for the ambiguity sets (derived from the measurements as in Fig. 3.1) and whose output consists of m lines representing the m binary functions representing the m faults. Therefore the fault isolation system can be considered as an ambiguity set-fault decoder. As such, the fault location task is accomplished using exactly the same binary logic obtained during isolation. However, there are various aspects of the

Table 4.1. Truth table of the binary functions representing the m faults.

	f_{1b}	f_{2b}	f_{3b}	...	f_{mb}
$v_b(1,1)$	1	0	0		1
$v_b(2,2)$	0	1	1		0
..					
..					
$v_b(1,L_1)$	0	0	0		0
$v_b(2,1)$	0	1	0		1
$v_b(2,2)$	1	0	0		0
..					
..					
$v_b(2,L_2)$	0	0	1		0
$v_b(3,1)$	0	0	1		0
$v_b(3,2)$	0	0	0		1
..					
$v_b(3,L_3)$	1	1			0
..					
..					
$v_b(n,1)$	0	1	0		1
$v_b(n,2)$	1	0	0		0
..					
..					
$v_b(n,L_n)$	0	0	1		0

ATE hardware implementation, one of which will be considered in a comparison of the storage requirements against the minimum squared deviation algorithm.

If the realization of the functions $\{f_{jb}\}$ was to be considered separately for every individual function, it would have been a straight forward sum of products (SOP) realization [9]. However, there are two problems involved that have to be simultaneously solved.

1. Finding minimal expressions for *all* the faults.
2. Assuring that there is no more than one fault condition having the same realization except to the degree of isolation required.

A simple hypothetical example will be first shown to clarify that the sum of product realization will tend to be a single product term which is not canonical (i.e. does not include all ambiguity set variables). However it includes a variable from every node. The significance of this will be discussed.

Example 4.1

Consider a 3-fault, 2-node case with each node having two ambiguity sets. Table 4.2 shows the reduced and complete truth tables of the problem. The only canonical SOP term which will make $f_{1b} = 1$ is $v_b(1,1).\bar{v}_b(1,2).\bar{v}_b(2,1).v_b(2,2)$ where the dot denotes logical AND and the bar denotes the complement. Because the ambiguity sets of any one node are mutually exclusive (i.e. only one of them can equal one), the following relations hold true:

$$v_{b(1,1)}.\bar{v}_b(1,2) = v_b(1,1) \tag{4.1}$$

$$\bar{v}_b(2,1).v_b(2,2) = v_b(2,2) \tag{4.2}$$

Hence

$$f_{1b} = v_b(1,1).v_b(1,2) \tag{4.3}$$

This expression in (4.3) is the same as one of the expressions obtained for f_1 using the classical minimization map as shown in Fig. 4.1.

It is clear that the realization which uniquely identifies every f_{jb} is in general the term represented by the product of the ambiguity sets including the fault, which can be obtained directly from the complete truth table (table 4.2) without referring to the map. This can be done if we consider only the prime implicant of f_j such that corresponding values for other functions are zero or "can't happen". Then use the relation

$$\bigcap_{\text{for all k and l}} v_b(i,k)\bar{v}_b(i,l) = \bigcap_{\text{for all k}} v_b(i,k) \tag{4.4}$$

Table 4.2. Truth table of example 4.1.

Reduced Truth Table of Example 4.1.

	f_{1b}	f_{2b}	f_{3b}
$v_b(1,1) = 1$	1	0	0
$v_b(1,2) = 1$	0	1	1
$v_b(2,1) = 1$	0	1	0
$v_b(2,2) = 1$	1	0	1

Complete truth table of example 4.1.

$v_b(2,2)$	$v_b(2,1)$	$v_b(1,2)$	$v_b(1,1)$	f_{1b}	f_{2b}	f_{3b}
0	0	0	0	x	x	x
0	0	0	1	x	x	x
0	0	1	1	x	x	x
0	0	1	1	x	x	x
0	1	0	0	x	x	x
0	1	0	1	x	x	0
0	1	1	0	0	1	x
0	1	1	1	x	x	x
1	0	0	0	x	x	x
1	0	0	1	1	0	x
1	0	1	0	x	x	1
1	0	1	1	x	x	x
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

x $\equiv$ can't happen

However, obtaining the complete truth table for a realistic large scale problem is not an easy matter. Meanwhile, the reduced table readily provides the means for finding a realization in a backwards way. We can use (4.4) such that the result is one for f_{jb} and zero for other functions. As we mentioned, each row in the reduced truth table assigns the value, one for the corresponding set without regard to other sets. The actual values of other sets are unimportant, e.g. the first term in the reduced table 4.2 means that if $v_b(1,1) = 1$ then $f_{1b} = 1$, $f_{2b} = 0$ and $f_{3b} = 0$ regardless of the values taken by $v_b(2,1)$ and $v_b(2,2)$. Of course, when we say $v_b(1,1) = 1$ we implicitly mean that $v_b(1,2) = 0$. Then $v_b(1,1)$ is the logical OR of the following two terms

$$\begin{aligned} & v_b(1,1)\overline{v}_b(1,2)v_b(2,1)\overline{v}_b(2,2) + v_b(1,1)\overline{v}_b(1,2)\overline{v}_b(2,1)v_b(2,2) \\ &= v_b(1,1)\overline{v}_b(1,2)[v_b(2,1) + \overline{v}_b(2,1)] \\ &= v_b(1,1)\overline{v}_b(1,2) \\ &= v_b(1,1) \qquad (4.5) \end{aligned}$$

This example demonstrates that the functions $\{f_{jb}\}$ can be realized using noncanonical product terms with the aid of the basic relation (4.4). Every product term realizing a function f_{jb} must include all variables corresponding to a prime implicant of the function. This implies that it includes an ambiguity set from every node encountered in the realization. If no attempt is made to reduce the number of nodes, the expression for every f_{jb} will be the product of all ambiguity sets containing the fault f_{jb}, which makes sense.

We emphasize that the product term is generally not the minimum. For example, f_{1b} in example 4.1 can be realized by $v_b(1,1)$ only, however the inclusion of the other node in other faults realizations will bring in a prime implicant for f_{1b} corresponding to $v_b(2,2)$ which cannot be ignored. In a practical sense, this means that if any node is to be measured, we have to make sure that the located fault exists in the correct set of the measured node, or an error has been made, inspite of the fact the fault can be realized using only one node or a smaller set of the measured nodes.

A similar situation exists in the fault codes derived in example 3.1 in section 3 where the last 4 faults can be identified using only the first node since these faults are the only faults which exist in the sets 3,4,5 and 6 of this node respectively. If some faith is put in these codes and if node measurements are to be done sequentially we can stop measurement, if the first node voltage range corresponds to one of the sets 3,4,5 or 6. If measurements are taken in parallel, then all sets of all measured nodes have to be checked to ensure correctness of the fault location.

4.2 Minimality of the Realization

The word minimality is rather a loose term that is often used to indicate different objectives [9]. We herein have a special feature that needs no additional testing cost if we add more variables in the realization so long as they all belong to the same node. Therefore, our objective is rather different from conventional senses of minimality. Meanwhile, the minimization has to be done for the multiple output realization (all f_{jb}) and not for a single function. Adding to this the large size of the problem, conventional minimization techniques applied to the truth table of $\{f_{jb}\}$ turn out to be inefficient.

The realization using the noncanonical product terms mentioned before will be used to uniquely realize all output functions (except for nonseparable faults). In the general case, this will yield neither the minimum number of nodes nor the minimum amount of hardware which is usually measured in terms of the number of gate inputs and the number of levels in the gate array [9]. However, the actual minimum may not be of a real value that is worth doing it. In fact, if any minimization is to be done, it should be for the number of test nodes to ensure convenience and quickness of the testing procedure. A special form of the minimization problem will now be considered to achieve this purpose, using what we call "separability" table which is different from the previous truth tables.

Consider Table 4.3 wherein the columns correspond to the fault pairs $\{f_j$ and $f_{j+1}\}$ and the rows correspond to the nodes $\{v_i\}$. An entry equal to 1 in the row node v_i means that the two faults corresponding to this entry lie in two different ambiguity sets of node v_i. A zero entry means that the fault pair is in the same set. If there is any pair of faults which cannot be split using any node, all the entries in the column corresponding to this pair will be zero. Such cases can be excluded from the table. Any realization of the functions $\{f_{(j)j+1}\}$ will not provide fault isolation (i.e. the decision rule for fault location). It will only provide a set of nodes which is enough for isolation. Once this set of nodes is found, we can go back to the truth table of the functions $\{f_{jb}\}$ to perform isolation. The separability table 4.3 can be considered as a truth table with inputs $\{v_i\}$ and outputs $\{f_{(j)j+1}\}$. The number of inputs is n. The missing (2^n-n) combinations really cannot happen because the complement of v_i has no meaning. However the realization desired is different from the conventional sense in that not all prime implicants need to be considered. Indeed any v_i corresponding to a prime implicant of a fault pair is an enough realization for this pair, i.e. it will split it. The smallest number of nodes which guarantee at least a single prime implicant for every fault pair is the required minimum. This can be obtained using the product term realizations then manipulating these terms to find the minimum set $\{v_i\}$ such that every term contains at least one node of this set. This will be demonstrated through an example.

Table 4.3. Separability table.

	f_{12}	f_{13}	f_{14}	..	f_{23}	f_{24}	..	$f_{(j)j+1}$	..	$f_{(m-1)m}$
v_1	1	0	0		0	0		1		0
v_2	1	0	0		0	0		1		0
v_3	1	1	1		1	1		0		1
..										
..										
v_n	0	0	0		0	0		1		1

	$\bar{v}_b(2,1)$		$v_b(2,1)$		
$\bar{v}_b(1,2)$	x	x	x	x	$\bar{v}_b(1,2)$
	x	x	x	0	$v_b(1,2)$
$v_b(1,1)$	x	x	x	x	
	x	1	x	x	$\bar{v}_b(1,2)$
	$\bar{v}_b(2,2)$	$v_b(2,2)$	$\bar{v}_b(2,2)$		

Fig. 4.1. Minimization map of f_{1b} in example 4.1.

$$
\begin{aligned}
f_{1b} &= v_b(1,1) \\
&= v_b(2,2) \\
&= \bar{v}_b(1,2) \\
&= \bar{v}_b(2,1) \\
&= v_b(1,1)v(2,2)
\end{aligned}
$$

Possible realizations of f_{1b}
using different one-zero
assignments of the can't happens.

Example 4.2

Consider the separability table 4.4 for a 5-node 5-fault case, each node having 2 ambiguity sets. The fault contents of the sets are shown in Table 4.5. The product term realizations for the 10 different combinations of fault pairs can be easily deduced from the separability table as follows:

$$f_{12} = v_4v_5$$

$$f_{13} = v_2v_3v_4$$

$$f_{14} = v_1v_3v_4v_5$$

$$f_{15} = v_1v_2v_3$$

$$f_{23} = v_2v_3v_5$$

$$f_{24} = v_1v_3$$

$$f_{25} = v_1v_2v_3v_4v_5$$

$$f_{34} = v_1v_2v_5$$

$$f_{35} = v_1v_4$$

$$f_{45} = v_2v_4v_5$$

The minimum number of nodes required for isolation is the number of nodes which if retained in the table while other nodes are excluded, a realization can still be obtained in the form of a product term for every fault pair. Some of the terms may include only a single node which is also a realization meaning that the corresponding fault pair cannot be split without this node. Finding the smallest set of nodes which will achieve this goal is not easy in the general case. However, the problem can be simplified by absorbing the redundancy in the expression via logical ORing of all the 10 expressions. This gives

$$\begin{aligned} K = {} & \underline{v_4v_5} + v_2v_3v_4 + \underline{v_1v_3v_4v_5} + \underline{v_1v_2v_3} + v_2v_3v_5 + v_1v_3 \\ & + \underline{v_1v_2v_3v_4v_5} + v_1v_2v_5 + v_1v_4 + v_2v_4v_5 \end{aligned} \tag{4.6}$$

Applying the following basic Boolean algebraic relation to the underlined terms [9]:

$$X + XY = X \tag{4.7}$$

we obtain the following simplification:

$$\begin{aligned} K &= \underline{v_4v_5} + v_2v_3v_4 + \underline{v_1v_2v_3} + v_2v_3v_5 + \underline{v_1v_3} + v_1v_2v_5 \\ &= v_1v_4 + \underline{v_2v_4v_5} \end{aligned} \tag{4.8}$$

Table 4.5. Ambiguity sets of example 4.2.

Set node	1	2
v_1	f_1,f_2,f_3	f_4,f_5
v_2	f_1,f_2,f_4	f_3,f_5
v_3	f_1,f_2	f_3,f_4,f_5
v_4	f_1,f_5	f_2,f_3,f_4
v_5	f_2,f_4	f_1,f_3,f_5

Table 4.4. Separability table of example 4.2.

	f_{12}	f_{13}	f_{14}	f_{15}	f_{23}	f_{24}	f_{25}	f_{34}	f_{35}
v_1	0	0	1	1	0	1	1	1	1
v_2	0	1	0	1	1	0	1	1	0
v_3	0	1	1	1	1	1	1	0	0
v_4	1	1	1	0	0	0	1	0	1
v_5	1	0	1	0	1	0	1	1	0

$$K = v_4v_5 + v_1v_3 + v_2v_3v_4 + v_2v_3v_5 + v_1v_2v_5 + v_1v_4$$

In a brute force manner we can tell that there are five possible solutions each provides 3 nodes if only retained in the expression of (4.8) each term will still have at least one variable (node). The solutions are (v_1,v_2,v_4), (v_1,v_3,v_4), (v_1,v_2,v_5), (v_1,v_3,v_5) and (v_3,v_4,v_5).

It is interesting to see that this problem has a graph theoretic equivalent known as the minimum feedback edge set problem [10]. The complexity of the computer implementation of the minimization algorithm is exponential in the number of variables [8].

4.3 Fault Isolation

Having obtained the minimum set of nodes, we can go back to the original truth table to find the SOP realization which will be a single product term including an ambiguity set from every node (after excluding redundant nodes). This realizations are guaranteed to identify every fault uniquely, i.e. there should be no more than one fault having the same expression except any pair of faults which may have been excluded before minimization.

There is an obvious effort in the process of minimization represented in forming the separability table then finding the minimum especially because the dimensionality of the separability table is larger than that of the original truth table. The number of fault pairs in m faults is $\frac{m(m-1)}{2}$ which is larger than m if m is greater than 3. If ensuring a minimal number of nodes is not a concern we can find the product term realizations directly from the original truth table and try simultaneously to solve the second problem mentioned in the beginning of this section. This means that we have to make sure that any realization we obtain for any of the faults is not a valid realization for any other fault. This can be done only if we operate on all faults simultaneously. For example if f_j is to be realized by $v(i_1,k_1)v(i_2,k_2)v(i_3,k_3)$, then f_j is equal to one if all these three sets are also equal to one. Then every other fault *must* be equal to zero when these sets are all equal to one, i.e. the product (logical ANDing) of the entries corresponding to these sets under every other fault *must* be zero. This means that to realize f_j and in the same time avoid valid realizations for other faults we will be taking the logical AND of the corresponding entries in the whole rows characterizing the ambiguity sets being considered which is nothing but taking the intersection of the sets, which has been used before. There are two ways for the intersection process to proceed:

1. Considering every fault f_{jb} separately and performing the intersection until we get a single nonzero element in the row. This element should of course correspond to f_{jb}. Then the process is repeated for all faults. The test nodes required in this case are the collection of all nodes realizing all faults.

2. Following the same objective set in section 3, i.e. intersecting the ambiguity sets in pairs and storing the result of intersection aiming at obtaining m rows each of them having a single nonzero element. This is exactly the same method followed in section 3 except that the intersection can now be obtained using logical ANDing of rows which may be stored in a single byte of storage.

We prefer the second way since it allows for wise selection of the sequence of sets to be intersected in order to approach the minimum by giving priority to the nodes having the maximum number of ambiguity sets, which we have also done before.

To complete the analogy between the binary and the multivalued forms we will list the dictionary in the form of binary expressions which can be realized by combinational logic.

4.4 Fault Location

As we mentioned before, the fault location is done directly using the binary expressions derived during the isolation process. The only thing needed in addition to that is to derive the binary input from voltage measurements.

Complexity of the algorithm: The intersection operation can now be done with great simplicity as the fault variables $\{f_{kb}\}$ will be stored as ones and zeros in one register (computer word) assuming that its length is greater than N_f. Simple logical ANDing of binary words will replace the intersection of ambiguity sets. The value of this simplification will be appreciated when the problem size gets larger.

A bound for the execution time can be easily derived in terms of a basic time unit T that takes the following two operations to be done.

i) Parallel ANDing of N_f pairs of bits, or simply ANDing of two computer words.
ii) Checking for null intersection, i.e., if the result of ANDing is zero or not. Referring to Fig. 3.2, we can deduce the following.
i) The time t of operations on two groups of sets whose numbers are L_1 and L_2 is $L_1 \times L_2 \times T$.
ii) At any intersection stage, the input number of sets in any of the two groups is less than N_f. Hence

$$t \leq \mathrm{Max}(L_i) \times N_f \times T \tag{4.9}$$

where $\mathrm{Max}(L_i)$ is the maximum number of sets in all nodes. *iii)* The number of intersection stages is at most $(N_v - 1)$. Then the overall execution time is bounded according to the inequality

$$t_{\mathrm{overall}} \leq \mathrm{Max}(L_i) \times N_f \times (N_v - 1) \times T \tag{4.10}$$

Added to this is, of course, the time needed in other manipulations like incrementing indices, etc.

Firmware: There are various possible ways of building an ATE for trouble shooting based on the logical isolation algorithm. One way is by using a digital comparator to compare the code generated from the test measurements against other codes in the dictionary. Another way is by converting the codes to their equivalent binary expressions and evaluating these expressions using a programmable gate array, for instance, with inputs derived from the ambiguity ranges of the test measurements. For example, the third entry in the dictionary (12211) corresponding to fault 6 has the binary equivalent.

$$f_{6b} = v_b(1,1)v_b(2,2)v_b(3,2)v_b(4,1)v_b(5,1)$$

If the RHS expression evaluates to one, fault 6 will be actual fault. Similar expressions hold for the rest of the faults. As such, the ATE can be viewed as an ambiguity set-fault decoder. The binary inputs $v_b(i,j)$ are derived from measurement of v_i according to relation 8. An interesting feature is that the input to the ATE does not need A/Dconversion but could be derived from the measured voltages using analog comparators as shown in Fig. 3.1.

At any case, the voltage ranges have to be stored in the ATE together with the fault codes. It is interesting to compare the storage requirements of this algorithm against those of the least squares method. The latter needs the storage of the fault-node voltage table which requires a storage of capacity S_1 given by

$$\begin{aligned} S_1 &= N_v N_f \quad \text{real numbers} \\ &= 2N_v N_f \quad \text{integers} \end{aligned}$$

The logical isolation requires the storage of the two tables of voltage ranges and fault codes which requires a capacity S_2 bounded by the inequality

$$\begin{aligned} S_2 &\le \Big[N_v \text{ x } 2 \text{ x } \mathrm{Max}(L_i)\Big] \text{ real numbers} \\ &+ \Big[N_v \text{ x } N_f\Big] \text{ integers} \\ &= N_v\Big[N_f + 4\,\mathrm{Max}(L_i)\Big] \text{ integers} \end{aligned}$$

S_2 grows as the number of faults while S_1 grows as twice this value.

5. ISOLATION WITH MULTIPLE TEST INPUT CONDITIONS

The effect of some faults may not be observable on the test nodes under normal d.c. operating conditions, or equally ambiguously some faults may have the same effect. This may sometimes be overcome by altering the normal d.c. inputs during testing procedure to allow the effect of faults to propagate to the test nodes, hoping to be able to distinguish ambiguous faults. The design of the test inputs is another problem which outside the scope of this manuscript. The extension of the algorithm to handle multiple input conditions is demonstrated in the next example.

Example 5.1

In [4] 20 faults in the video amplifier of Fig. 2.2 have been simulated using SYSCAP circuit simulator. The contents of the ambiguity sets of 10 initially chosen test nodes are shown in table 2.1. The nominal ambiguity sets of all nodes had been ignored with the belief that they do not help in isolation. It may seem that a node voltage having the nominal value is useless in diagnosis. However, this nominal response of some nodes in a faulty circuit gives information which is made use of in logical isolation.

With two inputs applied, every node acts like two noes in the one nominal input case insofar as the amount of information is concerned. This effectively yields the same problem with double the number of nodes. However, the sense of redundancy now has been slightly modified. The redundancy remains associated with the number of nodes rather than the number of measurements which in this case is equal to $N_i N_v$, where N_i is the number of inputs. Since the measurements of the chosen nodes will be done under every input condition, it is of no importance that some node gives redundant information with a particular input so long as its response with other inputs saves adding extra nodes. This would imply that before the intersection process is done, the maximum isolation capability has to be determined for every node. This can be obtained by considering every node separately and intersecting its ambiguity sets under all input conditions.

In the example of table 2.1, we will first ignore nodes 36 and 18 whose response is always indiscriminative. The rest eight nodes, when considered individually under the two input conditions, yield the following count of ambiguity sets:

Node:	11	8	5	2	27	26	33	16
No. of sets:	6	6	6	5	3	3	3	4

The intersection operations can now proceed as in the single input case to determine a group of nodes which will isolate the 20 faults with no redundancy. The sequence of nodes considered in intersection is also taken in a descending order of the number of sets, as it is intuitive that a node having a larger number of sets will have better isolation capability. The process is started by nodes 11 and 8 yielding 11 new sets. The result of intersecting the rest of the nodes is as follows:

Nodes	No. of sets
11,8	11
11,8,5	15
11,8,5,2	17
11,8,5,2,16	19
11,8,5,2,16,27	19
11,8,5,2,16,27,26	19
11,8,5,2,16,27,26,33	19

It is clear that nodes 27, 26 and 33 do not improve the isolation, stopping at only 19 sets which means that there is a pair of faults which have not been isolated among the original 20 faults.

There are two possible ways the fault dictionary can be formed and stored in the ATE.

1. If the two input case is viewed as two separate single input cases, then a fault dictionary may be generated for each case on the basis that the two inputs need not necessarily be both applied during the test. The fault codes for the two cases are shown in Table 5.1. The test should be first performed with the first input applied. If there is any ambiguity it will be then resolved by applying the second input. It is clear of course that faults which are ambiguous with the first input are isolated with the second one with the exception of the pair 10 and 12. The reverse is also true. As an example, suppose that the fault code is 25242 with the first input applied. The fault is then one of the group (1, 2, 6, 5, 8, 9, 11, 13, 14, 17 and 18). If the second input is applied and the code becomes 21111, the actual fault must be fault 2. Obviously if the second input was only applied, fault 2 would have been directly located. The input to be applied first is naturally the one that isolates more faults, which is input 2 in this case.
2. If the two input conditions are to be both applied before fault location is attempted, the two dictionaries have to be combined. The combined fault codes which isolate 18 faults are shown in table 5.2 where the indices of $v_{i,n}$ indicate the node and the input respectively. This of course increases both the storage requirements and the effort needed in fault location especially if it is done visually. This difficulty in handling the test manually when the dictionary size gets bigger may require fully automated trouble shooting aids based on the binary representation. At any case,

Table 5.1. Separate input fault codes of example 5.1.

INPUT1

FAULT CODE

VT11	VT8	VT5	VT2	VT16	Fault	
1	1	1	1	1	3	
1	2	1	1	1	7	
1	5	1	1	1	6	
1	5	2	2	1	19	
2	3	1	1	1	15	
2	4	2	4	2	16	
2	5	2	3	2	20	
2	5	2	4	1	10	12
2	5	2	4	2	1	2 4 5 8 9 11 13 14 17 18

INPUT2

FAULT CODE

VT11	VT8	VT5	VT2	VT16	Fault	
1	1	1	1	1	2	
2	1	1	1	1	5	
3	1	1	1	1	13	
4	2	5	2	3	14	
5	1	1	1	1	4	
5	2	2	1	1	9	
5	2	3	1	1	17	
5	2	4	2	3	18	
5	2	5	1	1	8	
5	2	5	2	2	11	
5	2	5	2	3	1	3 6 7 10 12 15 16 19 20

COMBINED INPUTS

V11,1	V11,2	V8,1	V8,2	V5,1	V5,2	V2,1	V2,2	V16,1	V16,2	Fault	
1	5	1	2	1	5	1	2	1	3	3	
1	5	2	2	1	5	1	2	1	3	7	
1	5	5	2	1	5	1	2	1	3	6	
1	5	5	2	2	5	2	2	1	3	19	
2	1	5	1	2	1	4	1	2	1	2	
2	2	5	1	2	1	4	1	2	1	5	
2	3	5	1	2	1	4	1	2	1	13	
2	4	5	2	2	5	4	2	2	3	14	
2	5	3	2	1	5	1	2	1	3	15	
2	5	4	2	2	5	4	2	2	3	16	
2	5	5	1	2	1	4	1	2	1	4	
2	5	5	2	2	2	4	1	2	1	9	
2	5	5	2	2	3	4	1	2	1	17	
2	5	5	2	2	4	4	2	2	3	18	
2	5	5	2	2	5	3	2	2	3	20	
2	5	5	2	2	5	4	1	2	1	8	
2	5	5	2	2	5	4	2	1	3	10	12
2	5	5	2	2	5	4	2	2	2	11	
2	5	5	2	2	5	4	2	2	3	1	

however, the voltage ranges of the ambiguity sets have to be stored for all input conditions in order to determine the set indices.

6. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

The method for generating an analog fault dictionary described in this report provides an efficient tool for fault diagnosis by logical means. The fault isolation algorithm presented in section 3 can be easily implemented with some extra software added to the circuit simulator. The execution time of the fault isolation routine, was shown to be bounded according to the analysis in section 4. Compared to the time needed for fault simulation, the logical fault isolation algorithm needs a trivial length of time. All this analysis is done once for all in the pretest stage. The outcome of logical fault isolation should be as follows:

1. A set of nodes whose d.c. voltage is to be measured during the test, and should be sufficient to achieve the required level of ambiguity in locating the actual fault after the test is done. This set of nodes is minimal in the sense that the prescribed degree of isolation cannot be obtained if any of these nodes is dispensed away with, which implies that none of the nodes is redundant. The term degree of isolation could roughly mean the number of faults within which the actual fault is ambiguous.
2. A table of voltage ranges for every node in the resulting test nodes, where every range is given an integer number that identifies an ambiguity group of faults any of which will cause the node voltage to be in the corresponding range. This means that the node voltage is effectively quantized into discrete ranges or multivalued logical levels. This quantization is expected to be always possible so long as the topology of the unit under test is such that the simulated faults are clustered in a way that all faults in a cluster have almost the same effect on a single test measurement. Of course, these clusters are different for different nodes.
3. A table of fault codes where every code identifies a fault or a group of faults up to the acceptable degree of isolation. Every code consists of integer numbers derived from those numbers assigned to the voltage ranges. The code length (number of integers in the code) is equal to the number of test nodes.

These two tables mentioned above constitute the required fault dictionary which has to be stored in the ATE. It has been shown in Section 4 that the storage requirements of the logically based analog fault dictionary are less than those of the least square based dictionary. The accompanying hardware is also much easier to implement. For large scale circuits, these advantages are very well appreciable.

The dictionary approach in general, like all other fault diagnosis approaches, cannot be claimed to be free from uncertainty which is introduced by statistical variations in component values and consequently of the circuit response. The efficacy of the dictionary also depends on how exhaustive the fault list is. Even with these drawbacks a dictionary for hard faults can be very efficient in diagnosing a large percentage of faults. The best reliability is achieved when the dictionary is used in conjunction with soft fault diagnosis routines, like multifrequency diagnosis, whenever it can be afforded. This is not always possible of course. For example, in field repairs where computing facilities are not available, a fault dictionary would be indispensable.

It may be sometimes possible to locate the fault with only fewer measurements among all measurements to be fed to the ATE. In such cases, completing the dictionary measurements may or may not be necessary depending on the level of confidence put in the dictionary. However, completing the measurements is preferable to make sure that the located fault does produce the coded response as stored in the ATE. If the test measurements are all done before attempting to locate the fault, this point will not be of importance. It is only meaningful if the test measurements are taken sequentially. In some other cases, if the code derived from measurements does not match with any other code in the dictionary a strange fault will be acknowledged. This is advantageous compared to other methods which will simply take the nearest fault on a least squares basis and declare it to be the actual fault.

For multiple test inputs, there are two options in compiling and using the fault dictionary.

1. It can be regarded as several single input cases. Then a fault dictionary may be generated for each test input on the basis that all inputs need not necessarily be applied during the test.
2. If all inputs are to be all applied before fault location is attempted, the dictionaries have to be combined as has been shown in section 5. This will generally produce longer fault codes and more storage will be needed. The first approach is recommended.

It is believed that the following points are important to be investigated to improve the performance of a logical fault dictionary:

1. A complete hardware implementation of an ATE based on logical isolation of faults. Several points related to this have been already discussed, e.g. storage requirements, interfacing, etc.. For flexibility and possible modifications in the dictionary, a microprocessor based ATE would naturally be most appropriate.
2. Investigating the effect of the components' statistical variation on the circuit response and consequently on the dictionary.

3. The design of test inputs for a better degree of isolation. The piecewise linear model and the formulation of the circuit equations into a complementary problem form readily provide the ground for this.

References

1. R. Duhamel and J. Rault, "Automatic Test Generation Techniques for Analog Circuits and Systems," *IEEE Trans. on Circuits and Systems,* Vol. CAS-26, pp. 411-440, July 1979.

2. W. A. Plice, "Overview of Current Automated Analog Test Design," *Proc. of the 1979 IEEE Test Conference,* Cherry Hill, New Jersey, pp. 128-136, September 1979.

3. R. Saeks, S. P. Singh and R. Liu, "Fault Isolation via Component Simulation," *IEEE Trans. on Circuit Theory,* Vol. CT-19, pp. 634-640, Nov. 1972.

4. W. Hochwald and J. Bastian, "A DC Approach for Analog Fault Dictionary Determination," *IEEE Trans. on Circuits and Systems,* Vol. CAS-26, pp. 523-529, July 1979.

5. A. T. Johnson, Jr., "Efficient Fault Analysis in Linear Analog Circuits," *IEEE Trans. on Circuits and Systems,* Vol. CAS-26, pp. 475-484, July 1979.

6. P. M. Lin, "DC Fault Diagnosis Using Complementary Pivot Theory," *Proc. 1982 IEEE Int'l. Symp. on Circuits and Systems,* pp. 1132-1135, May 1982.

7. C. Timaztepe and N. Prywes, "Generation of Software for Computer Controlled Test Equipment for Testing Analog Circuits," *IEEE Trans. on Circuits and Systems,* Vol. CAS-26, pp. 537-548, July 1979.

8. Narsingh Deo, "Graph Theory with Applications to Engineering and Computer Science," Prentice-Hall, 1974.

9. E. McCluskey, "Introduction to the theory of switching circuits," McGraw Hill, 1965.

10. LEMPEL, A., "Minimum Feedback Arc and Vertex Sets of a Directed Graph," *IEEE Trans. on Circuit Theory,* Vol. CT-13, pp. 399-403, Dec. 1966.